<http://blog.csdn.net/gaofuqi/article/details/46417281>

Spring 配置多个数据源，并实现动态切换

2015年06月09日 09:02:34

 40535

1.配置两个不同的数据源，如下（由于项目使用的是druid数据库连接，配置可以会复杂点比较）：

**[html]** [view plain](http://blog.csdn.net/gaofuqi/article/details/46417281#) [copy](http://blog.csdn.net/gaofuqi/article/details/46417281#)

1. <!-- 数据源配置1 -->

2. <bean id="testDataSource1" class="com.alibaba.druid.pool.DruidDataSource" init-method="init" destroy-method="close">

3. <property name="driverClassName" value="${db.driver}" />

4. <property name="url" value="${unity.db.jdbc.url}" />

5. <property name="username" value="${db.login.name}"></property>

6. <property name="password" value="${db.login.password}" />

7. <property name="filters" value="${db.filters}"></property>

8. <property name="maxActive" value="${db.pool.maxActive}"></property>

9. <property name="initialSize" value="${db.pool.initialSize}"></property>

10. <property name="minIdle" value="${db.pool.minIdle}"></property>

11. <property name="maxWait" value="${db.maxWait}"></property>

12. <property name="timeBetweenEvictionRunsMillis" value="${db.timeBetweenEvictionRunsMillis}"></property>

13. <property name="minEvictableIdleTimeMillis" value="${db.minEvictableIdleTimeMillis}"></property>

14. <property name="validationQuery" value="${db.validationQuery}"></property>

15. <property name="testWhileIdle" value="${db.testWhileIdle}"></property>

16. <property name="testOnBorrow" value="${db.testOnBorrow}"></property>

17. <property name="testOnReturn" value="${db.testOnReturn}"></property>

18. <property name="poolPreparedStatements" value="${db.poolPreparedStatements}"></property>

19. <property name="maxOpenPreparedStatements" value="${db.maxOpenPreparedStatements}"></property>

20. <!-- 监控数据库 -->

21. <property name="proxyFilters">

22. <list>

23. <ref bean="log-filter" />

24. </list>

25. </property>

26.

27. </bean>

**[html]** [view plain](http://blog.csdn.net/gaofuqi/article/details/46417281#) [copy](http://blog.csdn.net/gaofuqi/article/details/46417281#)

1. <!-- 数据源配置2 -->

2. <bean id="testDataSource2" class="com.alibaba.druid.pool.DruidDataSource" init-method="init" destroy-method="close">

3. <property name="driverClassName" value="${db.driver}" />

4. <property name="url" value="${pub.db.jdbc.url}" />

5. <property name="username" value="${db.login.name}"></property>

6. <property name="password" value="${db.login.password}" />

7. <property name="filters" value="${db.filters}"></property>

8. <property name="maxActive" value="${db.pool.maxActive}"></property>

9. <property name="initialSize" value="${db.pool.initialSize}"></property>

10. <property name="minIdle" value="${db.pool.minIdle}"></property>

11. <property name="maxWait" value="${db.maxWait}"></property>

12. <property name="timeBetweenEvictionRunsMillis" value="${db.timeBetweenEvictionRunsMillis}"></property>

13. <property name="minEvictableIdleTimeMillis" value="${db.minEvictableIdleTimeMillis}"></property>

14. <property name="validationQuery" value="${db.validationQuery}"></property>

15. <property name="testWhileIdle" value="${db.testWhileIdle}"></property>

16. <property name="testOnBorrow" value="${db.testOnBorrow}"></property>

17. <property name="testOnReturn" value="${db.testOnReturn}"></property>

18. <property name="poolPreparedStatements" value="${db.poolPreparedStatements}"></property>

19. <property name="maxOpenPreparedStatements" value="${db.maxOpenPreparedStatements}"></property>

20. <!-- 监控数据库 -->

21. <property name="proxyFilters">

22. <list>

23. <ref bean="log-filter" />

24. </list>

25. </property>

26.

27. </bean>

2.定义一个类继承AbstractRoutingDataSource实现determineCurrentLookupKey方法，该方法可以实现数据库的动态切换，如下：

**[java]** [view plain](http://blog.csdn.net/gaofuqi/article/details/46417281#) [copy](http://blog.csdn.net/gaofuqi/article/details/46417281#)

1. public class DynamicDataSource extends AbstractRoutingDataSource {

2. @Override

3. protected Object determineCurrentLookupKey() {

4. return DataSourceContextHolder.getDataSourceType();

5. }

6. }

3.定义一个可以设置当前线程的变量的工具类，用于设置对应的数据源名称：  
**[java]** [view plain](http://blog.csdn.net/gaofuqi/article/details/46417281#) [copy](http://blog.csdn.net/gaofuqi/article/details/46417281#)

1. public class DataSourceContextHolder {

2. private static final ThreadLocal<String> contextHolder = new ThreadLocal<String>();

3. /\*\*

4. \* @Description: 设置数据源类型

5. \* @param dataSourceType 数据库类型

6. \* @return void

7. \* @throws

8. \*/

9. public static void setDataSourceType(String dataSourceType) {

10. contextHolder.set(dataSourceType);

11. }

12.

13. /\*\*

14. \* @Description: 获取数据源类型

15. \* @param

16. \* @return String

17. \* @throws

18. \*/

19. public static String getDataSourceType() {

20. return contextHolder.get();

21. }

22.

23. /\*\*

24. \* @Description: 清除数据源类型

25. \* @param

26. \* @return void

27. \* @throws

28. \*/

29. public static void clearDataSourceType() {

30. contextHolder.remove();

31. }

32. }

然后在spring中配置，如下：

**[html]** [view plain](http://blog.csdn.net/gaofuqi/article/details/46417281#) [copy](http://blog.csdn.net/gaofuqi/article/details/46417281#)

1. <!-- 编写spring 配置文件的配置多数源映射关系 -->

2. <bean class="com.sino.access.database.DynamicDataSource" id="dataSource">

3. <property name="targetDataSources">

4. <map key-type="java.lang.String">

5. <entry value-ref="testDataSource1" key="<span style="font-family: Arial, Helvetica, sans-serif;">testDataSource1</span><span style="font-family: Arial, Helvetica, sans-serif;">"></entry></span>

6. <entry value-ref="testDataSource2" key="testDataSource2"></entry>

7. </map>

8. </property>

9. <property name="defaultTargetDataSource" ref="testDataSource1">

10. </property>

11. </bean>

</bean>  
这样配置两个数据源对应的key分别为testDataSource1和testDataSource2，默认数据库是testDataSource。  
4.完成以上步骤后，如果没有数据库的事务管理，已经可以实现数据库的动态切换了。但是如果涉及到数据库的事务管理，需要在数据库事务开启切换数据库，  
否则数据库的切换只能在下次数据库操作时才生效。可以定义一个aop处理类在数据库事务开启之前切换数据库，如下：

**[java]** [view plain](http://blog.csdn.net/gaofuqi/article/details/46417281#) [copy](http://blog.csdn.net/gaofuqi/article/details/46417281#)

1. public class DataSourceAspect implements MethodBeforeAdvice,AfterReturningAdvice

2. {

3.

4. @Override

5. public void afterReturning(Object returnValue, Method method,

6. Object[] args, Object target) throws Throwable {

7. // TODO Auto-generated method stub

8. DataSourceContextHolder.clearDataSourceType();

9. }

10.

11. @Override

12. public void before(Method method, Object[] args, Object target)

13. throws Throwable {

14.

15. if (method.isAnnotationPresent(DataSource.class))

16. {

17. DataSource datasource = method.getAnnotation(DataSource.class);

18. DataSourceContextHolder.setDataSourceType(datasource.name());

19. }

20. else

21. {

22. DataSourceContextHolder.setDataSourceType(SinoConstant.DataSourceType.unityDataSource.toString());

23. }

24.

25. }

26. }

5.设置数据库事务切面和切换数据库切面执行的顺序，如下：  
**[html]** [view plain](http://blog.csdn.net/gaofuqi/article/details/46417281#) [copy](http://blog.csdn.net/gaofuqi/article/details/46417281#)

1. <aop:config>

2. <aop:pointcut id="transactionPointCut" expression="execution(\* com.test.service.\*.\*(..))" />

3. <aop:advisor pointcut-ref="transactionPointCut"

4. advice-ref="txAdvice" order="2" />

5. <aop:advisor advice-ref="dataSourceExchange" pointcut-ref="transactionPointCut" order="1"/>

6. </aop:config>

利用aop的order属性设置执行的顺序，这样实现了带事务管理的spring数据库动态切换。

由于之前博客少了部分代码，自己重新写了一个demo：

链接: https://pan.baidu.com/s/1c1FurkO 密码: qiec